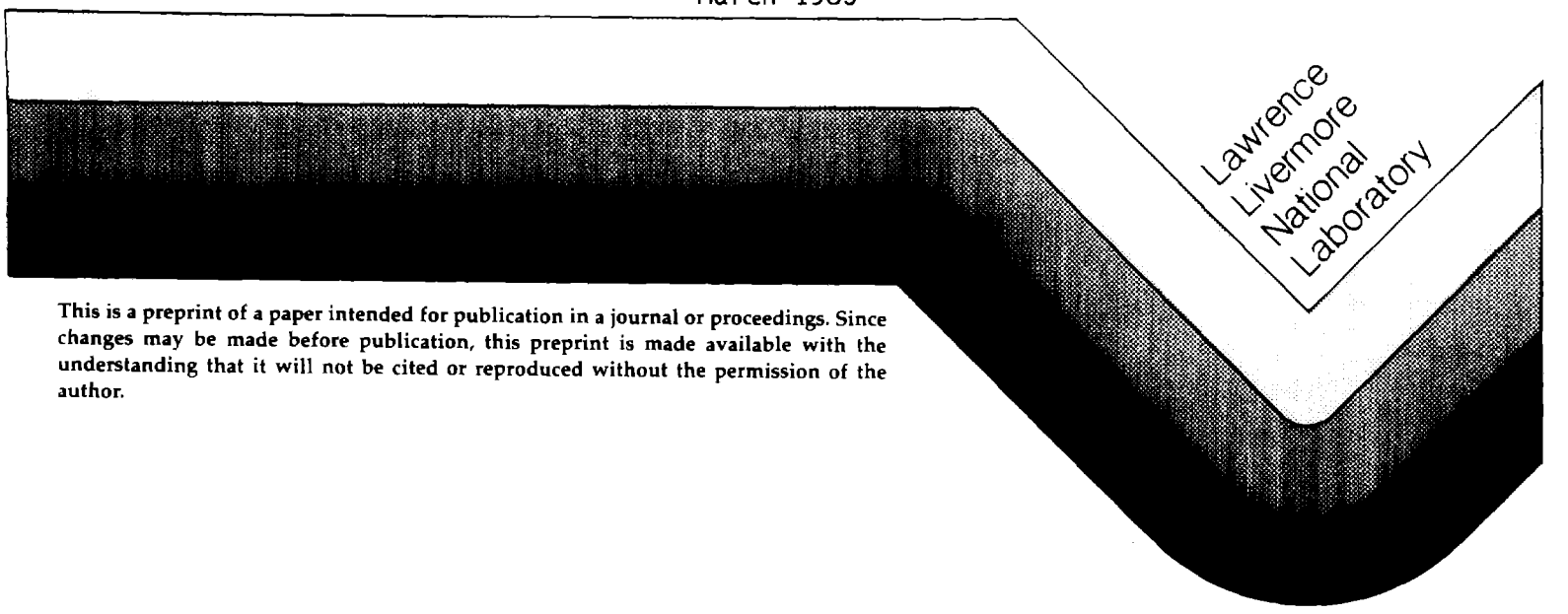
UCRL- 92312
PREPRINT

TRANSFORM DOMAIN ADAPTIVE FILTERING USING A RECURSIVE DFT

G. A. Clark
Lawrence Livermore National Laboratory
M. A. Soderstrand
T. G. Johnson
Signal and Image Processing Laboratory
University of California
Department of ECE
Davis, CA 95616

This paper was prepared for submittal to
1985 International Symposium on Circuits and Systems
Kyoto, Japan - June 5-7, 1985

March 1985



TRANSFORM DOMAIN ADAPTIVE FILTERING USING A RECURSIVE DFT*

G. A. Clark

University of California
Lawrence Livermore National Lab
P.O. Box 808, L-156
Livermore, CA 94550

M. A. Soderstrand
T. G. Johnson
Signal and Image Processing Laboratory
University of California
Department of ECE
Davis, CA 95616

ABSTRACT

This paper discusses the use of a recursive DFT (discrete Fourier transform) with frequency-domain adaptive filters. We show that, even though the recursive DFT ordinarily has stability problems when implemented in finite precision arithmetic, it is stable when used with an appropriate numerical roundoff procedure, making it practical for use with frequency-sampling type adaptive filters.

INTRODUCTION

The activity in frequency-domain adaptive filtering research has produced several algorithms in recent years (see the references in [1,2]). Most of them use the DFT to process the data in blocks (rather than one sample at a time), and perform fast convolutions/correlations. On the other hand, the frequency sampling-type adaptive filters (FSAFs) process the data one sample at-a-time, using the DFT not for fast convolution purposes, but to provide an orthogonal transformation that allows for rapid convergence of the filter weights when appropriate gradient step size parameters are employed [3-6]. The disadvantage of the FSAFs is that they require more computational complexity than the LMS adaptive filter and all of the block-type filters.

The purpose of this paper is to discuss the use of the recursive DFT [7,8] with the FSAFs. The advantages of the recursive DFT are that it produces a new set of Fourier coefficients at every time sample, it preserves the convergence speed benefits of orthogonal transformation, and it requires less computational complexity than the fast Fourier transform (FFT). The disadvantage is that the recursive DFT ordinarily has stability problems when implemented using finite precision arithmetic [7,8]. We show, however, that if an appropriate roundoff procedure is used, the recursive DFT is stable, making it practical for use with FSAFs.

FREQUENCY-SAMPLING ADAPTIVE FILTERS

The basic frequency sampling adaptive filter of interest here is the one described in [3-6] and depicted in Fig. 1. Most of the authors use an FFT to implement the orthogonal transform, but [3] appears to use a recursive DFT, without addressing the effects of finite precision arithmetic. The filter equations are summarized as follows:

The complex-valued input signal x(n) is passed through a delay line and a DFT to form the vector

$$\underline{s}_n = [s_0(n),\ s_1(n),\ \ldots\ ,\ s_{N-1}(n)]^T \qquad (1)$$

where n is the discrete time index, N is the number of filter weights, and superscript "T" represents matrix transposition. The complex-valued filter weight vector is:

$$\underline{w}_n = [w_0(n),\ w_1(n),\ \ldots\ ,\ w_{N-1}(n)]^T \qquad (2)$$

The output y(n) is given by

$$y(n) = \underline{w}_n^T\, \underline{s}_n = \underline{s}_n^T\, \underline{w}_n \qquad (3)$$

The complex weight adjustment algorithm is

$$\underline{w}_{n+1} = \underline{w}_n + A\, \underline{s}_n^*\, e(n) \qquad (4)$$

where superscript "*" denotes complex conjugation, α is a positive convergence constant, d(n) is the desired response, e(n) = d(n) - y(n), and A is an NxN diagonal matrix, the elements of which are given by:

$$a_k(n) = \frac{\alpha}{\hat{\sigma}^2_{s,k}(n)}\ , \quad k = 0,1,\ldots,N-1 \qquad (5)$$

The time-varying input power estimates $\hat{\sigma}^2_{s,k}(n)$ are computed recursively by:

$$\hat{\sigma}^2_{s,k}(n+1) = \beta\hat{\sigma}^2_{s,k}(n) + (1-\beta)\, s_k(n)\, s_k^*(n) \qquad (6)$$

where $0 < \beta < 1$ is the so-called smoothing constant.

*Work performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under contract number W-7405-ENG-48.

Relationships among the various frequency-domain filters

It was shown in [2] that the block adaptive filter provides a general framework, of which all of the frequency domain adaptive filters are special cases. Particularly interesting is the recent result of [9], showing that the FSAF of [4,5] is a special case of the block adaptive filter when the block length L=1. Since the LMS filter of Widrow [1,2] is equal to the block LMS filter when L=1, the FSAF should be equivalent to the LMS filter when a constant convergence factor is used.

THE RECURSIVE DFT

The Fourier coefficients in Eq. (1) can be written as follows:

$$s_k(n) = \sum_{i=0}^{N-1} x(n-i)e^{-j\frac{2\pi}{N}ik},$$

$$k = 0,1,\ldots,N-1 \qquad (7)$$

where k is the discrete frequency index. Note that an entire N-point DFT is computed at each time instant in sliding window fashion. One can derive the following recursion for this DFT [10]:

$$s_k(n) = e^{-j\frac{2\pi}{N}k}\, s_k(n-1) + x(n) - x(n-N) \qquad (8)$$

It can be shown that Eq. (8) yields the frequency sampling filter structure [4-6] depicted in Fig. 1.

The recursive DFT implements poles on the unit circle, which are used to cancel the zeros of the comb filter $1 - z^{-N}$. With infinite precision arithmetic, the poles on the unit circle cause no stability problems, but with finite precision, the poles can easily be located outside the unit circle, due to quantization effects. This problem is typically handled by sampling the system transfer function on a circle of radius r, where r is very slightly less than unity (typically $r = 1 - 2^{-26}$) [7,8]. This approach is mentioned in [3]. In this case, Eq. (8) becomes:

$$s_n(k) = re^{-j\frac{2\pi}{N}k}\, s_{n-1}(k) + x(n) - r^N x(n-N) \qquad (9)$$

Unfortunately, the use of r can cause the recursive DFT to decay with time, giving unsatisfactory spectrum estimates. In the next section, we show that a stable recursive DFT can be found (without the use of r) by using a simple roundoff scheme.

NUMERICAL STABILITY OF THE RECURSIVE DFT

Hardware implementations of the recursive DFT have two main problems: (1) they cannot implement poles exactly on the unit circle, so errors accumulate over time, and (2) they cannot implement zeros which exactly cancel the poles. In floating-point hardware, the source of these problems is primarily the adders, and in fixed point hardware, the problem source is the multipliers.

Floating-Point Implementation

Here, scaling is necessary before addition to equalize the exponents of the numbers to be added. This means that one number gets truncated or rounded to a word size consistent with the other. This can cause pole and zero locations to be moved away from the unit circle. Consider the zero frequency term (k = 0) of the recursive DFT:

$$s_0(n) = s_0(n-1) + x(n) - x(n-N) \qquad (10)$$

$s_0(n)$ should always equal the sum of the last N inputs x(n) through x(n-N+1). To work properly, x(n-N) must exactly remove the x(n) added N samples previously. The error occurs because the floating-point processor scales x(n) and x(n-N) differently, due to the varying nature of $s_0(n-1)$. To correct for this, we have devised the following algorithm that forces x(n) and x(n-N) to be scaled the same:

$$s_n(0) = s_{n-1}(0) + [x(n) + c] - [x(n-N) + c] \qquad (11)$$

where c is a positive constant chosen such that x(n) and x(n-N) are consistently rounded to the same number of significant digits. A general discussion of these issues is omitted for brevity, but it is important to note that floating point realizations are susceptible to finite precision difficulties.

Fixed-Point Implementation

Here, the additions are exact as long as the inputs are scaled to prevent overflow, so only the multipliers cause problems. For analysis purposes, we assume the following: (1) all additions are exact (with zero error), (2) all data are represented by b-bit two's complement arithmetic, and (3) the products $s_k(n)\, w_k(n)$ are computed by ROM table lookup and the result is rounded to b bits (see Fig. 2).

Under these assumptions, the rounding algorithm becomes the only source of problems because it determines whether or not the poles lie on the unit circle. If simple truncation is used, the effective pole locations are inside the unit circle, causing long-term decay in the spectrum estimates. Our goal is to develop a rounding algorithm that will effectively locate the poles on the unit circle. With the hardware of Fig. 2, we find that simple two's complement rounding accomplishes this goal. It locates the poles sometimes inside and sometimes outside the unit circle, ensuring that signal components at the pole frequency do not experience long-term decay or growth. To illustrate this, we simulated the hardware of Fig. 2 using both truncation and

rounding. We let $x(n) = 100\ \delta(n)$, where $\delta(n)$ is the unit sample. If we let $F_k(n)$ denote the exact (infinite precision) value of the kth coefficient of the N-point DFT at time n, the error is:

$$E_k(n) = |F_k(n)| - |s_k(n)| \qquad (12)$$

The error accumulation is demonstrated in Fig. 3a for the truncation case, and Fig. 3b for the rounding case. Note that rounding produces small, bounded, stable errors.

COMPUTATIONAL COMPLEXITY

The most significant advantage of the recursive DFT is its associated computational complexity savings and hardware simplifications. As an example, we designed hardware implementations of a fast Fourier transform (FFT) and the recursive DFT. We used 8-bit words, 256x8 ROMs for the multiplication, and standard 8-bit adders. A summary of the hardware requirements is given in Table I, showing that the recursive DFT offers considerable hardware savings. Furthermore, if one processing unit is multiplexed to obtain the N Fourier coefficients, the recursive DFT operates faster than the FFT. Thus, even though we can trade hardware complexity against speed for either circuit, the recursive DFT is always more efficient than the DFT.

Table I. Hardware computational complexity for the FFT and recursive DFT.

	Number of Adders		Number of ROMs	
	FFT	Rec. DFT	FFT	Rec. DFT
N	$(5N/2)\log_2 N$	4N+1	$2N\log_2 N$	4N
8	60	33	48	32
16	160	65	128	64
32	400	129	320	128
64	960	257	768	256
128	2240	513	1792	512
256	5120	1025	4096	1024
512	11520	2049	9216	2048
1024	25600	4097	20480	4096

CONCLUSIONS

We have shown that the recursive DFT provides computational savings, and can achieve stable spectrum estimates for fixed-point implementations when appropriate rounding is applied. Our current work involves applying the recursive DFT to frequency sampling adaptive filters.

REFERENCES

1. G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," IEEE Trans. Circuits Syst., Vol. CAS-28, pp. 584-592, June 1981, and IEEE Trans. Acoust., Speech, Signal Processing, Joint Special Issue on Adaptive Signal Processing, Vol. ASSP-29, pp. 744-752, June 1981.

2. G. A. Clark, S. R. Parker, and S. K. Mitra, "A unified approach to time-and-frequency domain realization of FIR adaptive digital filters," IEEE Trans. Acoust., Speech, and Sig. Proc., Vol. ASSP-31, No. 5, Oct. 1983, pp. 1073-1083.

3. R. R. Bitmead and B. D. O. Anderson, "Adaptive frequency sampling filters," IEEE Trans. Circuits Syst., Vol. CAS-28, pp. 524-533, June 1981, and IEEE Trans. Acoust., Speech, Signal Processing, Joint Special Issue on Adaptive Signal Processing, Vol. ASSP-29, pp. 684-694, June 1981.

4. S. Shankar Narayan and A. M. Peterson, "Frequency domain least-mean-square algorithm," Proc. IEEE, Vol. 69, pp. 124-126, Jan. 1981.

5. S. S. Narayan, A. M. Peterson, and M. J. Narasimha, "Transform domain LMS algorithm," IEEE Trans. Acoust., Speech, and Signal Processing, Vol. ASSP-31, No. 3, June 1983, pp. 609-615.

6. J. C. Lee and C. K. Un, "On the convergence behavior of frequency-domain LMS adaptive digital filters," Proc. IEEE Intl. Conf. Acoustics, Speech, and Signal Processing, San Diego, CA, March 19-21, 1984, pp. 46.4.1-46.4.4.

7. B. Gold and C. M. Roder, Digital Processing of Signals, McGraw-Hill, 1969.

8. A. V. Oppenheim and R. W. Schafer, "Digital Signal Processing," Engelwood Cliffs, NJ, Prentice-Hall, 1975.

9. J. C. Lee and C. K. Un, "A Reduced Structure of the Frequency-Domain Block LMS Adaptive Digital Filter," Proc. IEEE, Vol. 72, No. 12, December 1984, pp. 1816-1818.

10. M. A. Soderstrand and T. G. Johnson, "Hardware Implementations of a Moving Discrete Fourier Transform," Proc. ISMM 1984 Conference on Mini and Micro Computers, Ceasar's Palace, Las Vegas, NV, December 1984, pp. 151-153.

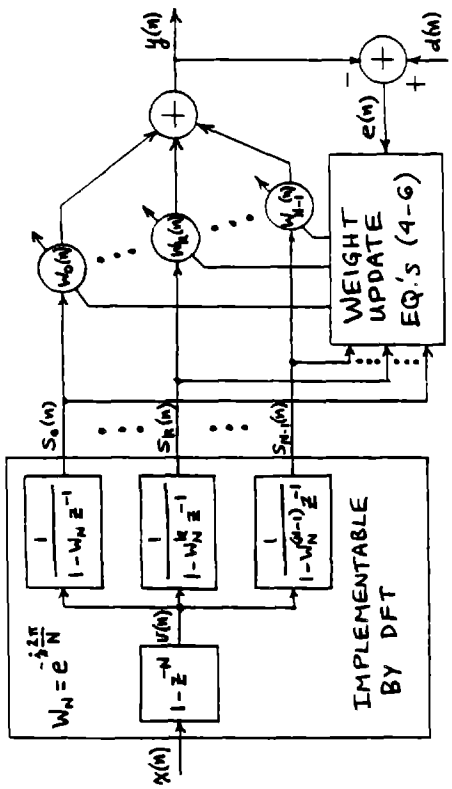


Fig. 1. The frequency-sampling adaptive filter using the recursive DFT.

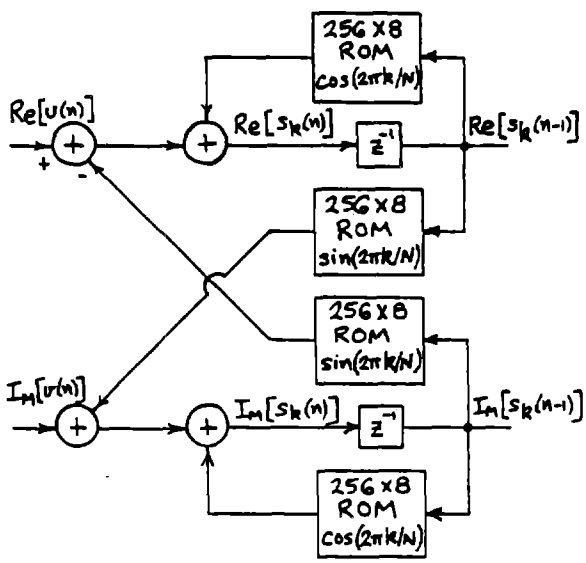


Fig. 2. A fixed-point hardware implementation of the kth resonator of the recursive DFT structure.

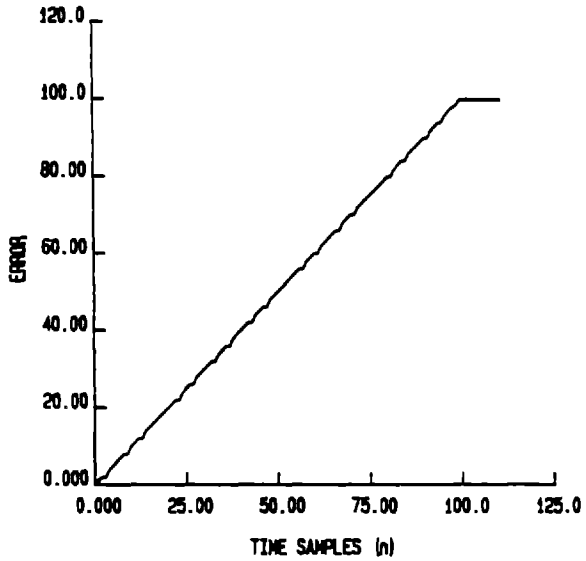


Fig. 3a. The error $E_k(n)$ in the first (k=1) recursive DFT coefficient, for standard two's complement arithmetic with truncation. Note that the error grows with time and saturates at 100, the value of the spectral component at k=1.

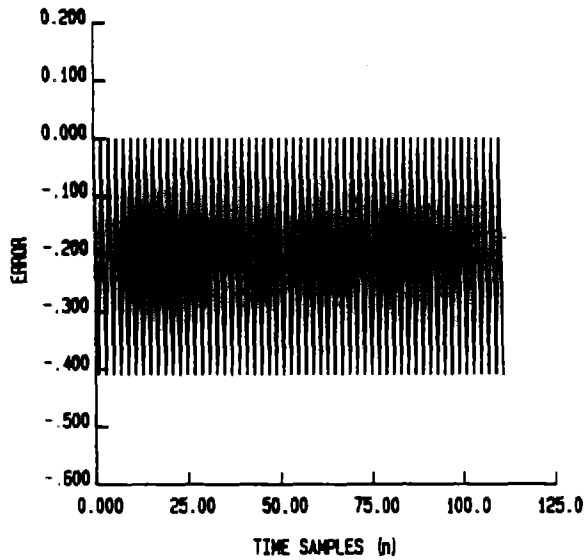


Fig. 3b. The error $E_k(n)$ in the first (k=1) recursive DFT coefficient, for standard two's complement arithmetic with rounding. Note that the error is small, bounded, and stable.